ارائه یک الگوریتم ترکیبی برای زمان بندی وظایف جریان کاری علمی روی بستر ابر محاسباتی با محدودیت مهلت زمانی و هدف کاهش هزینه اجرا

ملیحه حریری[1]*، مصطفی نوری بایگی[2] و سعید ابریشمی [3]
[1] دانشجوی کارشناسی ارشد مهندسی کامپیوتر دانشگاه فردوسی مشهد، *malihe.hariri@mail.um.ac.ir*
[2] استادیار دانشگاه فردوسی مشهد ، *nouribaygi@um.ac.ir*
[3] استادیار دانشگاه فردوسی مشهد ، *s-abrishami@um.ac.ir*

چکیده : جریان‌های کاری علمی برای پردازش داده‌های عظیم و تحلیل‌ها و شبیه‌سازی‌های پیچیده استفاده می‌شوند، درنتیجه نیازمند منابع محاسباتی قدرتمندی هستند که بتوانند نتایج موردنظر را در زمان قابل‌قبول و با هزینه مطلوب تولید کنند. به‌این منظور از منابع زیرساخت‌های توزیع‌شده‌ای چون ابر محاسباتی به دلیل مزایایی ازجمله دسترسی به منابع مجازی ، نامحدود و کشسان برای اجرای جریان‌های کاری، استفاده می‌شود و وظایف جریان‌های کاری برای اجرا روی منابع محاسباتی نگاشت می‌شود. برای نگاشت وظایف به منابع محاسباتی، مسئله به‌صورت مسئله زمان‌بندی مدل می‌شود؛ اما مسئله زمان‌بندی با وجود محدودیت‌ها و معیارهای مختلف یک مسئله **NP-hard** است. به‌همین جهت الگوریتم‌های مختلفی برای حل مسئله زمان‌بندی در زمان چندجمله‌ای ارائه شده‌است. الگوریتم ارائه‌شده در این پژوهش یک الگوریتم ترکیبی مبتنی بر مدل ریاضی است که مسئله زمان‌بندی را با شکستن مسئله به زیرمسئله‌های کوچک‌تر و زمان‌بندی بسته‌هایی از وظایف و مدل کردن مسئله با استفاده از یک مدل ریاضی خطی- صحیح انجام می‌دهد. مزیت این روش کاهش هزینه اجرای وظایف در یک مهلت زمانی مشخص نسبت به الگوریتم‌های زمان‌بندی ارائه‌شده است.

کلید واژه‌ها: *زمان‌بندی وظایف، الگوریتم ترکیبی، مدل ریاضی، مهلت زمانی، هزینه*

۱- مقدمه

جریان‌های کاری مجموعه‌ای از وظایف محاسباتی هستند که بین وظایف آن‌ها وابستگی‌های داده‌ای وجود دارد. پردازش و اجرای جریان های کاری در سال های اخیر روی بستر ابر محاسباتی موردتوجه محققان زیادی قرار گرفته است. برای زمان‌بندی جریان‌های کاری الگوریتم‌های مختلفی ارائه شده است که زمانبندی وظایف را روی منابع ابر محاسباتی به گونه‌ای انجام می‌دهند که نیازهای عملیاتی و غیرعملیاتی مطرح شده در مسأله برآورده شود.

در مسائل زمان‌بندی یک مسأله مهم انتخاب نوع و تعداد منابع مناسب برای زمان‌بندی وظایف است. انتخاب منابع مناسب تأثیر زیادی در عملکرد الگوریتم زمان‌بندی دارد. در مسائل زمان‌بندی اهداف و معیارهای مختلفی درنظر گرفته می‌شود که در اغلب مسائل زمان و هزینه به عنوان دو معیار مهم درنظر گرفته شده‌اند. انتخاب منابع سریع‌تر هزینه زیادی را بر کاربر تحمیل می‌کند و استفاده از منابع کندتر با وجود اینکه هزینه را کاهش می‌دهد ولی قادر به اجرای وظایف در زمان ارائه شده نیست، بنابراین باید بین این دو توازن برقرار شود. الگوریتم ارائه شده در این مقاله به چالش‌های ذکر شده به خوبی پاسخ می‌دهد. نوآوری‌های ارائه شده در این الگوریتم را می‌توان به صورت زیر بیان کرد:

۱. ارائه یک روش مناسب برای توزیع مهلت زمانی با استفاده از مدل خطی برای فراهم‌آوری منابع مناسب
۲. ارائه یک الگوریتم زمان‌بندی با مدل ریاضی خطی-صحیح برای زمان‌بندی با کیفیت بیش‌تر روی منابع فراهم‌شده

۲- کارهای پیشین

برای مسأله زمان‌بندی دسته‌بندی‌های مختلفی ارائه شده است که در [۱]، [۲]، [۳]، [۴] و [۵] مسأله زمان بندی از منظرهای مختلف مورد بررسی قرار گرفته است. در [۱] مسأله زمان‌بندی از دیدگاه‌های مختلفی موردبررسی قرار داده است. از نظر معیارهای زمان‌بندی می‌توان مقالات را به چندین بخش تقسیم کرد. اغلب الگوریتم‌های زمان‌بندی، دو فاکتور زمان اجرا و هزینه اجرای جریان کاری روی منابع محاسباتی را در نظر می‌گیرند و تلاش می‌کنند، با در نظر گرفتن یک معیار به‌عنوان محدودیت مسأله زمان‌بندی، معیار دیگر را بهینه کنند؛ اما علاوه بر این دو معیار، معیارهای دیگری از جمله قابلیت اعتماد، مصرف

انرژی، امنیت، بهره‌وری منابع و بارکاری نیز درنظر گرفته‌می‌شود.

از نظر نگاشت وظایف جریان‌های کاری به منابع ابر محاسباتی، می‌توان کارهای انجام‌شده را به سه صورت ایستا، پویا و ترکیبی تقسیم‌بندی کرد. در این کارها با توجه به سیاست مسأله، نگاشت وظایف به منابع به سه روش ایستا، پویا و ترکیبی انجام می‌پذیرد.

در [۶] Moa و همکاران یک الگوریتم پویا برای زمان‌بندی وظایف جریان‌های کاری روی بستر ابر عمومی با محدودیت مهلت زمانی و هدف کاهش هزینه اجرا ارائه داده‌اند. در این کار برای بسته‌های وظایف آماده منابعی فراهم می‌شود که اجرای آن‌ها را قبل از مهلت زمانی و با کمترین هزینه به پایان برسانند.

در [۷]، Thai و همکاران دو روش زمان‌بندی ایستا مبتنی بر مدل ریاضی برای زمان‌بندی بسته‌هایی از وظایف ارائه داده‌اند. در روش اول یک مدل ریاضی برای تعیین نوع و تعداد منابع موردنیاز برای اجرای بسته‌های وظایف شامل چندین وظیفه، ارائه‌شده است، با این هدف که در مهلت زمانی تعیین‌شده به پایان برسد و هزینه اجرای وظایف روی منابع کاهش یابد. در روش دوم برای هریک از وظایف در بسته یک مدل ریاضی جدید ارائه می‌دهد که نوع منبع موردنظر برای اجرای هر وظیفه را به‌دست می‌آورد.

در [۸] Rodriguez و همکاران یک الگوریتم ابتکاری با زمان‌بندی ترکیبی برای جریان‌های کاری علمی در زیر ساخت ابر عمومی ارائه داده‌اند. در این کار در مرحله ایستا مهلت زمانی روی وظایف جریان کاری توزیع می‌شود و در مرحله پویا منابع موردنیاز برای زمان‌بندی وظایف آماده با استفاده از الگوریتم کوله‌پشتی فراهم می‌شود و زمان‌بندی وظایف روی منابع فراهم شده انجام می‌شود.

در [۹] Rodriguez و همکاران یک الگوریتم ترکیبی زمان‌بندی و فراهم‌آوری منابع ارائه داده‌اند، با این هدف که زمان اجرای جریان کاری کمینه شود و زمان‌بندی با بودجه موردنظر به پایان برسد. در مرحله ایستا بودجه روی وظایف جریان کاری توزیع می‌شود و منابع موردنیاز با استفاده از یک مدل ریاضی فراهم می شوند و زمان‌بندی وظایف آماده روی منابع فراهم شده انجام می‌شود.

۳- *سیستم پیشنهادی*

۳-۱- *مدل سامانه و فرضیات مسأله*

جریان کاری به‌صورت یک گراف جهت‌دار بدون دور مدل می‌شود که $T = \{t_1, t_2, \dots, t_n\}$ شامل مجموعه وظایف جریان کاری و E مجموعه یال‌های این گراف است. یک یال $e_{ij} = (t_i, t_{i+1})$ بین دو وظیفه t_i و t_{i+1} جود دارد، اگر بین این دو وظیفه وابستگی داده‌ای وجود داشته باشد. درنتیجه وظیفه فرزند تا قبل از اجرای کامل وظایف پدرانش نمی‌تواند شروع به اجرا کند.

در این مقاله منابع فراهم‌شده از فراهم‌کننده گوگل با دوره‌های زمانی یک دقیقه‌ای استفاده شده است و به ازای استفاده جزئی بیش از دوره زمانی، هزینه این زمان، به هزینه کلی اضافه می‌شود.

ظرفیت ذخیره‌ساز فرض شده در این مسأله بی‌نهایت درنظر گرفته‌شده‌است. داده‌های ورودی و خروجی هریک از وظایف، روی ذخیره‌ساز ذخیره می‌شوند و زمان انتقال داده از ذخیره‌ساز، در زمان اجرای وظیفه و زمان اجرای منبع درنظر گرفته‌شده‌است. همچنین از هزینه مالی استفاده از ذخیره‌ساز صرف‌نظر می‌شود. زمان فراهم‌آوری منابع در قالب هزینه در الگوریتم درنظر گرفته‌شده‌است. به‌این‌صورت که الگوریتم به نسبت افزایش سرعت منبع انتخابی به ارزان‌ترین منبع ، هزینه مازاد باید پرداخت کند. همچنین منابع، زمانی‌که به ۵ ثانیه آخر دوره زمانی خود می‌رسند در صورتی که هیچ وظیفه‌ای برای اجرا وجود نداشته‌باشد، خاموش می‌شوند.

۳-۲- *روش پیشنهادی*

در این مقاله، یک الگوریتم ترکیبی برای زمان‌بندی وظایف جریان‌های کاری علمی، ارائه شده‌است که درواقع بهبودی روی روش ارائه شده در [۸] است. روش پیشنهادی این مقاله از دو مرحله ایستا و پویا تشکیل شده است. در مرحله ایستا جریان کاری پیش‌پردازش می شود و ساختار خط لوله‌های جریان کاری مشخص می شود و با استفاده از یک روش بهبودیافته با استفاده از مدل ریاضی خطی، مهلت زمانی وظایف محاسبه می شود. در مرحله پویا زمان‌بندی بسته‌های وظایف آماده (خط لوله‌های آماده) با استفاده از یک روش فراهم‌آوری منابع با مدل برنامه‌ریزی خطی- صحیح انجام خواهد شد. در ادامه به تشریح هر دو مرحله پرداخته‌شده‌است.

۱-۱-۱- *مرحله ایستا*

در مرحله ایستا ابتدا ساختار خط‌لوله‌های جریان کاری مشخص می‌شود. نحوه تشکیل خط‌لوله‌های جریان کاری در الگوریتم ۱ نشان داده‌شده‌است.

تشکیل خط لوله‌ها به این صورت است که ابتدا وظایف به ترتیب توپولوژیکی مرتب می‌شوند، سپس خط لوله‌ها ساخته

الگوریتم ۱: پیدا کردن خط لوله‌ها

```
1. procedure findPipeline(Task t, Pipeline p, List<Task> ischeck){
2. if ( t.getChildList().size() > 1 OR  t.getChildList().size() == 0
   OR t.getChildList().get(0).getParentList().size() > 1) {
3.       if (!pipeline.isEmpty())
4.           t.putInPipeline(pipeline);
5.     return pipeline;
6. }
7. t.putInPipeline(pipeline);
8. return findPipline(t.getChildList().get(0), pipeline, ischeck);
```

می‌شوند. برای هر وظیفه مرتبی که پردازش نشده است الگوریتم تلاش می‌کند خط لوله‌ای بسازد که با این وظیفه شروع شود. بهترین حالت بازگشت وقتی اتفاق می‌افتد که وظیفه هیچ فرزندی نداشته باشد یا بیش از یک فرزند داشته باشد و یا چندین پدر داشته باشد. حالت بازگشتی زمانی اتفاق می‌افتد که وظیفه فقط یک فرزند و فقط یک پدر داشته باشد. در این حالت وظیفه به خط لوله اضافه می‌شود و بازگشت با فرزند آن ادامه پیدا می‌کند. وقتی خط لوله شناسایی شد و بازگشت به پایان رسید، این کار برای وظیفه پردازش نشده بعدی تکرار می‌شود تا زمانی‌که هیچ وظیفه پردازش نشده‌ای باقی نماند.

در مرحله بعد به هر وظیفه در جریان کاری یک مهلت زمانی تخصیص داده می‌شود. مهلت زمانی تخصیص داده‌شده به وظایف در الگوریتم زمان‌بندی و هزینه اجرا تأثیر زیادی دارد. به این منظور روش توزیع مهلت زمانی ارائه شده در مقاله [۸] را با استفاده از مدل ریاضی خطی بهبود بخشیدیم.

الگوریتم توزیع مهلت زمانی وظایف به صورت زیر عمل می‌کند:

برای توزیع مهلت زمانی وظایف، ابتدا زودترین زمانی که وظایف می‌توانند به پایان برسند، روی سریع‌ترین ماشین موجود مطابق

$$eft_t = max_{p\in t.parents}\{eft_p\} + Runtime_t^{fastestVm}$$

محاسبه می شود. سپس مقدار بیشینه زودترین زمان پایان بین تمام وظایف مطابق $eft_{max} = max_{t\in w}\{eft_t\}$ محاسبه می‌شود و میزان زمان اضافی از مهلت زمانی جریان کاری $sparetime = \delta_t - eft_{max}$ به‌دست می‌آید.

حال با استفاده از یک مدل ریاضی، زمان اضافی به وظایف جریان کاری با استفاده از فرمول‌های ۱ تا ۴ تخصیص داده می شود. در ادامه متغیرها و پارامترهای مدل ریاضی در جدول ۱ و جدول ۲ آورده شده‌است.

جدول ۱ متغیرهای مدل ریاضی

$spare_l$	زمان تخصیص داده شده به هر سطح از جریان کاری
T	متغیر تعریف شده برای تبدیل مدل به خطی

جدول ۲ پارامترهای ورودی

$max\, t_{i_l}$	بیشترین زمان اجرا وظایف هر سطح جریان کاری
$\sum t_{i_l}$	مجموع زمان اجرای همه وظایف جریان کاری
$SpareTime_w$	زمان اضافی باقی‌مانده از جریان کاری

$$Max\ T \quad (۱)$$

$$\frac{max\, t_i + spare_l}{\sum t_{il}} \geq T\ \ \forall l \in level \quad (۲)$$

$$\sum spare_l \leq sparetime_w\ \forall l \in level \quad (۳)$$

$$spare_l \geq 0 \quad (۴)$$

در این مدل، تابع هدف بیشینه کردن مقدار $\frac{max\, t_i + spare_l}{\sum t_{il}}$ به ازای هر سطح از جریان کاری است. با این محدودیت که میزان زمان اضافی داده شده به هر سطح گراف کمتر از میزان زمان اضافی کل جریان کاری باشد. بعد از حل مدل، مقدار $spare_l$ برای هر سطح از جریان کاری به‌دست می‌آید و به تمام وظایف هر سطح از گراف جریان کاری تخصیص داده می‌شود. سپس مهلت زمانی δ_t داده‌شده به هریک از وظایف جریان کاری به صورت فرمول ۵ محاسبه می‌شود.

$$\delta_t = max_{p\in t.parents}\{eft_p\} + sparetime_t + runtime_t^{fastestVm} \quad (۵)$$

۲-۱-۱- مرحله پویا

در این مرحله، زمان‌بندی وظایف آماده در صف اجرا آغاز می‌شود. برای زمان‌بندی وظایف داخل صف اجرا، ابتدا وظایف به بسته‌هایی از وظایف آماده یا بسته‌هایی از خط لوله‌های آماده، دسته‌بندی می‌شوند. اگر منابعی از قبل وجود داشته باشند که بخشی از دوره زمانی آن‌ها استفاده شده‌است و هم‌اکنون بی‌کار هستند، به‌اندازه تعداد وظایفی از بسته که منبع می‌تواند در مهلت زمانی مشخص شده و قبل از اتمام دوره زمانی‌اش اجرا کند، برای زمان‌بندی به منبع تخصیص داده می‌شود و برای مابقی وظایف باقی‌مانده در بسته، با استفاده از یک طرح فراهم‌آوری، منابع موردنیاز برای اجرای آن‌ها تعیین می‌شود.

برای فراهم‌کردن منابع موردنیاز برای وظایف باقی‌مانده از یک مدل ریاضی خطی-صحیح استفاده‌شده است که نوع و تعداد منابع موردنیاز برای اجرای بسته‌های وظایف (بسته‌های خط لوله) را به‌گونه‌ای تعیین می‌کند، که هزینه اجرای این بسته از وظایف کمینه باشد و قبل از مهلت زمانی وظایف به پایان برسد.

جدول ۳ : ویژگی‌های هر منبع

c_{vmt}	هزینه منبع در هر دوره پرداختی
p_t^{vmt}	زمان پردازش وظیفه‌ای با بزرگ‌ترین زمان پردازش روی منبع با نوع vmt

جدول ۴: متغیرهای مسأله

C_{bot}	هزینه اجرای یک بسته از وظایف
$Cost_{vm}$	هزینه استفاده از منبع برحسب تعداد دوره‌های زمانی مصرف‌شده
$N_{vmt,k}$	متغیر صحیح نشان‌دهنده تعداد وظایف داده‌شده به k امین نمونه نوع vmt
$U_{vmt,k}$	مدت‌زمانی که k امین نمونه از نوع vmt برای اجرای وظایف استفاده کرده است
$P_{vmt,k}$	متغیر صحیح نشان‌دهنده تعداد دوره‌های زمانی اجاره‌شده برای k امین نمونه نوع vmt
$L_{vmt,k}$	متغیر باینری که در صورتی‌که k امین نمونه از نوع vmt اجاره‌شده باشد مقدار ۱ می‌گیرد. در غیر این صورت مقدار ۰ دارد
n	تعداد وظایف موجود در بسته
δ_{bot}	مهلت زمانی تخصیص داده‌شده به بسته وظایف (بسته خط لوله‌ها)
$IntTol$	معیار توقف حل‌کننده
τ	دوره زمانی هر منبع محاسباتی (ثانیه)
c_{vmt}	هزینه اجاره یک دوره زمانی منبع نوع vmt

مدل ریاضی ارائه‌شده، نوع و تعداد منابع برای اجرای بسته‌های وظایف را به‌گونه‌ای انتخاب می‌کند که هزینه اجرای این بسته از وظایف روی منابع موجود کمینه شود و اجرای وظایف قبل از مهلت زمانی داده‌شده به پایان برسد. مدت‌زمانی که k امین نمونه از نوع vmt مشغول اجرا بوده، به صورت فرمول $U_{vmt,k} = p_t^{vmt} * N_{vmt,k}$ محاسبه می‌شود. هزینه هر نوع منبع استفاده شده بر اساس تعداد دوره‌های زمانی اجاره شده توسط منبع و هزینه ثابت منبع به ازای یک دوره زمانی به صورت فرمول $Cost_{vm} = p_{vmt,k} * c_{vmt}$ محاسبه می شود. بنابراین مسأله را می‌توان به‌صورت زیر فرموله کرد:

$$Min\ C_{bot} \qquad ۶$$

$$c_{bot} - cost_{vm} \geq 0 \qquad ۷$$

$$\sum_{j \in VMT} \sum_{k \in vm_j} N_{vmt,k} = n \qquad ۸$$

$$N_{vmt,k} \leq n * l_{vmt,k} \qquad ۹$$

$$\frac{U_{vmt,k}}{\tau} \leq P_{vmt,k} \qquad ۱۰$$

$$N_{vmt,k} \geq L_{vmt,k} \qquad ۱۱$$

$$\frac{U_{vmt,k}}{\tau} + (1 - IntTol) \geq P_{vmt,k} \qquad ۱۲$$

$$U_{vmt,k} \leq \delta_{bot} \qquad ۱۳$$

تابع هدف مسأله، کمینه کردن هزینه اجرای بسته وظایف است که در فرمول ۶ محاسبه شده است. محدودیت ۷ نشان می‌دهد که هزینه اجرای این بسته از وظایف باید نسبت به هزینه‌ی اجاره منبع در دوره‌های زمانی کمینه شود. محدودیت ۸ تضمین می‌کند، همه وظایف بسته روی منابع پردازش می شوند محدودیت ۹ و ۱۱ تضمین می‌کند به هر منبع فراهم شده، حتماً وظیفه‌ای برای اجرا داده خواهد شد. محدودیت ۱۰ و ۱۲ تعداد دوره‌های زمانی استفاده‌شده منبع را با محاسبه مدت‌زمان اجرای منبع و با گردکردن به نزدیک‌ترین عدد صحیح بزرگ‌تر به دست می‌آورد. محدودیت ۱۳ تضمین می‌کند، مدت‌زمان اجرای وظیفه روی منبع زمان‌بندی‌شده باید کمتر از مهلت زمانی داده شده باشد. بعد از حل مسأله متغیر $N_{vmt,k}$ تبدیل به یک طرح فراهم‌آوری منابع به‌صورت

$RP_{vmt} = (VMT, NumVm, NumTask)$ می‌شـــود. جواب به‌دســـت‌آمده تعیین می‌کند از هر نوع منبع VMT چه تعداد باید فراهم شـــود و هر منبع قادر به زمان‌بندی چند وظیفه اســـت. در الگوریتم ۲ شـــبه کد الگوریتم ترکیبی پیشـــنهادی ارائه‌شده است.

```
1.  for(all bot in BOT)
2.      if(bot.size()> 1)
3.          if(idleVmlist.isEpmty())
4.                  MilpResult = Milp.solveMilp(bot,deadlineBot);
5.                  for(all milpresult =(vmType,numVm,NTask)
6.                      MyTask t = getTaskFromBot();
1.                      submitTaskVm(t, vm);
2.                  end for
3.           end if
4.          else
5.                  MyVm vm = idleVmList.get();
6.                  if(vm.NTask != 0)
7.                       submitTaskVm(t, vm);
8.                  end if
9.          end else
10.     end if
11.     else
12.          if(idleVmList.isEpmty())
13.                 MyVM vm = provisionsingleBot(bot);
14.                 MyTask t = getTaskFromBot();
15.                 submitTaskVm(t, vm);
16.          end if
17.          else
18.                 MyVm vm = idleVmList.get();
19.                 if(vm.NTask!=0)
20.                    submitTaskVm(t, vm);
21.                 end if
22.          end else
23.     end else
24. end for
```

الگوریتم ۱ زمان‌بندی وظایف جریان کاری

۴- *ارزیابی کارایی*

در شبیه سازی‌های صورت گرفته شده از جریان‌های کاری علمی *Montage* و *Epigenomics* و *Sipht* و *Ligo* ســـتفـاده شده‌است و جریان‌های کاری با تعداد تقریباً ۱۰۰۰ وظیفه درنظر گرفته‌شـده‌اسـت، تخمین اندازه وظایف ۱۰۰٪ دقیق نیسـت و در محاســـبه اندازه وظایف تغییرات حدود ۱۰٪± با توزیع یکنواخت برای نرمال‌سازی زمان اجرای وظایف لحاظ می‌گردد.

در این مقاله برای هر جریان کاری ۷ مهلت زمانی درنظر گرفته‌شده‌است. سخت‌ترین مهلت زمانی δw_1 برابر مجموع زمان اجرای وظایف روی مسیر بحرانی و زمانی‌است که برای خواندن و نوشتن از ذخیره‌ساز نیاز است. مابقی مهلت‌های زمانی به صورت فرمول‌های ۱۴ تا ۲۰ محاسبه می‌شود.

$\delta w_{int} = \delta w_1/2$	۱۴
$\delta w_2 = \delta w_1 + \delta w_{int}$	۱۵
$\delta w_3 = \delta w_2 + \delta w_{int}$	۱۶
$\delta w_4 = \delta w_3 + \delta w_{int}$	۱۷
$\delta w_5 = 4 * \delta w_1$	۱۸
$\delta w_7 = 8 * \delta w_1$	۱۹
$\delta w_7 = 10 * \delta w_1$	۲۰

تابع درنظر گرفته‌شـــده برای اجرای جریان‌های کاری از منابع فراهم‌کننده گوگل با دوره زمانی ۶۰ ثانیه فرض شده‌است و ۴ نوع از این منابع درنظر گرفته‌شـــده‌اســـت که در جدول قابل‌مشـــاهده اســـت. همچنین ذخیره‌ســـاز با بیش‌ترین ســـرعت خواندن و نوشتن فرض شده است.

نام	حافظه	*Mips*	قیمت بر هزینه
N1-standard-1	*3.75 GB*	*2.75*2750*	***$0.00105000***
N1-standard-2	*GB 7.5*	*5.5*2750*	***$0.00210525***
N1-standard-4	*GB 15*	*11*2750*	***$0.004221026***
N1-standard-8	*GB 30*	*22*2750*	***$0.00846315***

روش پیشـــنهادی این مقاله الگوریتم *MHPSLP* اســـت که با کار ارائه شـــده در [۸] از نظر زمان و هزینه موردارزیابی قرار گرفت است.

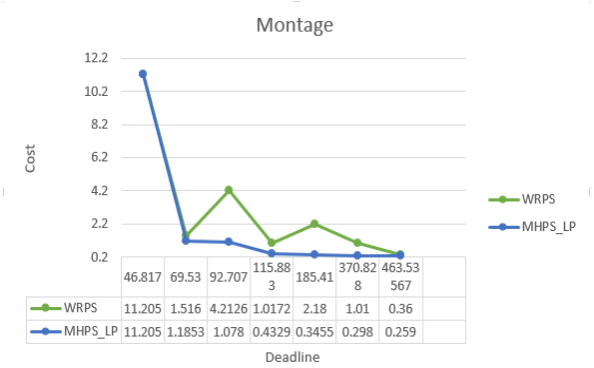
۴-۱- *نتایج آزمایش‌ها و تحلیل‌ها*

- جریان کاری *Montage*

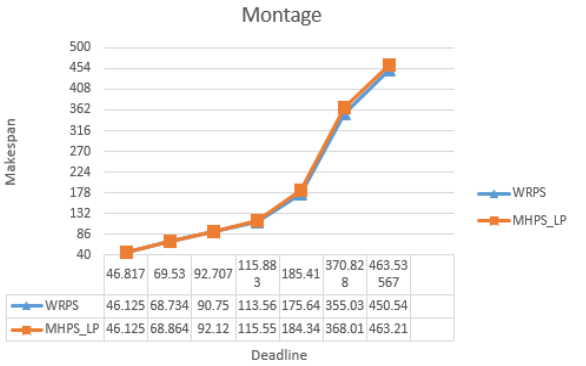
در این جر یان کاری هزینه و ز مان اجرا روی الگوریتم پیشـنهادی و الگوریتم موردمقایسـه در شـکل ۱و شـکل ۲، قابل مشـاهده اسـت. در الگوریتم *WRPS* با افزایش مهلت زمانی در برخی موارد هزینه افزایش پیدا کرده اســـت که به دلیل توزیع نامناســـب مهلت زمانی با روش *pspare* اســـت، که روند کاهش هزینه اجرای جریان‌های کاری در الگوریتم پی شنهادی مقاله قابل مشاهده است.

در شـــکل۲ زمان اجرای الگوریتم‌ها روی جریان کاری *Montage* نشان داده شده‌است. در الگوریتم *MHPSLP*، برای کم کردن هزینه بهترین استفاده از زمان موجود را نسبت به الگوریتم

موردمقایسه داشته‌اند. این بهبود در مهلت‌های زمانی بالاتر در الگوریتم پیشنهادی ما قابل مشاهده است.



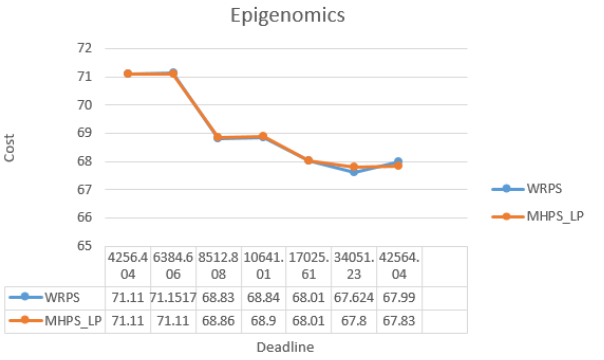
شکل ۱: زمان اجرای جریان کاری Montage



شکل ۲: هزینه اجرای جریان کاری Montage

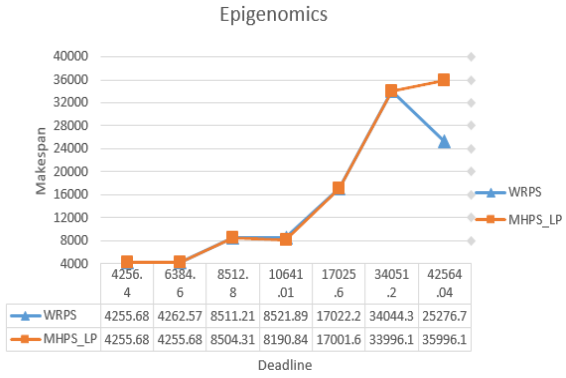
- جریان کاری Epigenomics

هزینه و زمان اجرا مقایسه در شکل ۳ و شکل ۴، نشان داده شده است. در الگوریتم *WRPS* در برخی مهلت‌های زمانی هزینه افزایش یافته است که به دلیل توزیع نادرست مقدار زمانی اضافی جریان کاری بین وظایف است. در الگوریتم *MHPSLP* این مشکل برطرف شده‌است، اما در مهلت زمانی سوم تا پنجم هزینه نسبت به الگوریتم موردمقایسه بیشتر شده است.



شکل ۳: هزینه اجرای جریان کاری Epigenomics

زمان اجرای الگوریتم‌ها روی جریان کاری *Epigenomics* درشکل ۴ نشان داده شده است. در الگوریتم *WRPS* زمان اجرای جریان کاری در مهلت‌های زمانی سوم تا پنجم کاهش یافته‌است و این به دلیل ساختار این جریان کاری و فرضی است که برای فراهم‌آوری منابع بسته های خط لوله های وظایف درنظر گرفته‌شده‌است. ما در این مسأله فرض کردیم که اندازه زمان اجرای همه خط لوله‌ها به اندازه خط لوله‌ای است که بیش‌ترین زمان اجرا را دارد، در نتیجه الگوریتم از زمان باقی‌مانده تا مهلت زمانی به‌خوبی استفاده نمیکند و زمان اجرا کاهش می‌یابد و از طرفی هزینه نیز افزایش پیدا می‌کند. این مشکلات در الگوریتم *MHPSLP* در همین مهلت‌های زمانی و به همین شکل دیده می‌شود.



شکل ۴: زمان اجرای جریان کاری Epigenomics

- جریان کاری *Ligo*

در این جریان کاری هزینه اجرا و زمان اجرای جریان کاری در نمودارهای ۵ و ۶ آمده‌است. در این جریان کاری هزینه اجرا در الگوریتم *MHPSLP* کاهش یافته است و منابع می‌توانند چندین وظیفه را زمان‌بندی و به ترتیب اجرا کنند در الگوریتم *WRPS* به علت مهلت زمانی کمی که به وظایف با زمان اجرای بالا داده است مجبور می‌شود از منابع سریع‌تر برای اجرا استفاده کند، درنتیجه هزینه بالاتر می‌رود.

	178.21 32	264.67 31	352.89 74	441.12 18	705.79 26	1411.7 905	1764.4 873
WRPS	8.8355	7.6676	6.07	6.0466	4.89	4.46	4.56
MHPS_LP	8.8355	7.32909	6.0466	5.956	4.81	4.38	4.42

شکل ۵: هزینه اجرای جریان کاری Ligo

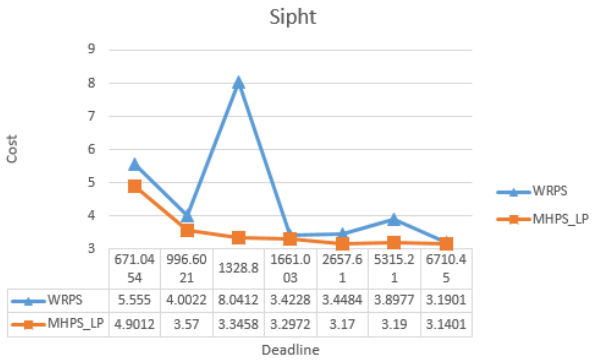
نتایج ارزیابی زمان اجرای الگوریتم‌ها روی جریان کاری *Ligo* نشان می‌دهد، عملکرد الگوریتم *MHPSLP* در استفاده از زمان باقی‌مانده بهتر از الگوریتم‌های دیگر است. مخصوصاً در مهلت‌های زمانی بالاتر این تفاوت واضح‌تر می شود که نشان‌دهنده عملکرد بهتر الگوریتم در مهلت‌های زمانی بالاتر است.

	178.2 132	264.6 731	352.8 974	441.1 218	705.7 926	1411. 79	1764. 487
WRPS	176.511	262.99	352.29	352.958	704.47	1411.36	1516.6
MHPS_LP	177.511	264.019	352.88	401.98	705.81	1411.64	1755.69

شکل۶: زمان اجرای جریان کاری Ligo

- جریان کاری *Sipht*

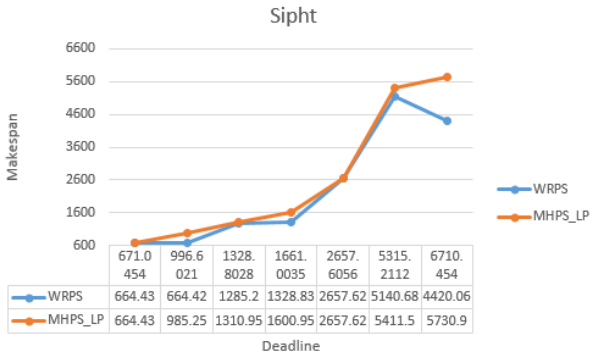
در این جریان کاری هزینه اجرا و زمان اجرای جریان کاری در شکل ۷ و شکل ۸ نشان داده شده است. در این جریان کاری هزینه اجرا اهمیت توزیع مهلت زمانی روی وظایف را در این جریان کاری به وضوح بیش‌تری نسبت به جریان‌های کاری بررسی‌شده نشان می‌دهد که به دلیل ساختار متفاوت این جریان کاری و تفاوت زمان اجرای بسته‌های وظایف است.



	671.04 54	996.60 21	1328.8	1661.0 03	2657.6 1	5315.2 1	6710.4 5
WRPS	5.555	4.0022	8.0412	3.4228	3.4484	3.8977	3.1901
MHPS_LP	4.9012	3.57	3.3458	3.2972	3.17	3.19	3.1401

شکل ۷: *هزینه اجرای جریان کاری Sipht*

نتایج ارزیابی زمان اجرای الگوریتم‌ها روی جریان کاری *Sipht* در شکل ۸ نشان داده شده است. در این جریان کاری یک بسته از وظایف با زمان اجرای بالا وجود دارد که الگوریتم *WRPS* توجهی به اهمیت آن نمی‌کند و ارزان‌ترین منابع را برای آن انتخاب می‌کند.



	671.0 454	996.6 021	1328. 8028	1661. 0035	2657. 6056	5315. 2112	6710. 454
WRPS	664.43	664.42	1285.2	1328.83	2657.62	5140.68	4420.06
MHPS_LP	664.43	985.25	1310.95	1600.95	2657.62	5411.5	5730.9

شکل۸: زمان اجرای جریان کاری Sipht

۵- نتیجه‌گیری

در این مقاله یک الگوریتم ترکیبی بهبود یافته برای زمان بندی وظایف جریان های کاری در ابر محاسباتی ارائه شده است. در این الگوریتم هدف کمینه کردن هزینه اجرای وظایف روی منابع ابر محاسباتی است، به این شرط که اجرای همه وظایف جریان کاری در مهلت زمانی تعیین شده به پایان برسد. بنابراین مهلت زمانی تخصیص داده شده به وظایف تأثیر زیادی روی عملکرد الگوریتم زمان‌بندی خواهد داشت.

الگوریتم توزیع مهلت زمانی در[۸] مهلت زمانی وظایف را به درستی توزیع نمی‌کند. از طرفی به دلیل درنظر گرفتن زمان فراهم‌آوری منابع، زمان زیادی را از دست می‌دهد. به‌همین‌دلیل الگوریتم زمان‌بندی *WRPS* قادر به اجرای موفق و کامل همه

وظـایف برخی از جریـان‌هـای کـاری مثـل *Ligo* و *Sipht* در سـخت‌ترین مهلت زمانی نیسـت. از طرفی این توزیع نامناسـب مهلت زمانی باعث افزایش هزینه اجاره منابع در برخی موارد شده اسـت که به صـورت نمونه می‌توان به جریان کاری *Sipht* اشـاره کرد که در بخش قبل شـرح داده‌شـد. در اغلب جریان‌های کاری، کار ارائه شده ما عملکرد بهتری نسبت به الگوریتم *WRPS* داشته ا ست. جز در مواردی که ب ستگی به ساختار جریان کاری تغییراتی خلاف انتظار رخ داده اسـت، در کل می‌توان گفت مدل ریاضـی منابع را هوشمندانه‌تر انتخاب می‌کند و توزیع درست مهلت زمانی روی وظایف جریان کاری الگوریتم را در انتخاب منابع درسـت هدایت می‌کند. در آینده درنظر داریم مشکلات پیش‌رو را برطرف کنیم و فرضیات را به‌گونه‌ای تغییر دهیم که این مشکلات برطرف شود.

مراجع

[1] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," Concurr. Comput., vol. 29, no. 8, pp. 1–32, 2017.

[2] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," Futur. Gener. Comput. Syst., vol. 52, pp. 1–12, 2015.

[3] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities," Futur. Gener. Comput. Syst., vol. 50, pp. 3–21, 2015.

[4] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: a survey," J. Supercomput., vol. 71, no. 9, 2015.

[5] L. Thai, B. Varghese, and A. Barker, "A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds," Futur. Gener. Comput. Syst., vol. 82, pp. 1–11, 2018.

[6] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," Proc. 2011 Int. Conf. High Perform. Comput. Networking, Storage Anal. - SC '11, p. 1, 2011.

[7] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," Futur. Gener. Comput. Syst., vol. 29, no. 1, pp. 158–169, 2013.

[8] M. A. Rodriguez and R. Buyya, "A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds," in Proceedings of the International Conference on Parallel Processing, 2015, vol. 2015-Decem, pp. 839–848.

[9] M. A. Rodriguez and R. Buyya, "39 Budget-Driven Resource Provisioning and Scheduling of Scientific Workflow in IaaS Clouds with Fine-Grained Billing Periods," vol. 9, no. 4, pp. 1–22, 2017.